Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-16 (2022) 344–349

Four MPC implementations compared on the Quadruple Tank Process Benchmark : pros and cons of neural MPC ⋆

Pierre Clément Blaud * **Philippe Chevrel** * **Fabien Claveau** *
Pierrick Haurant ** **Anthony Mouraud** ***

* *IMT Atlantique, LS2N, UMR CNRS 6004, F-44307 Nantes, France (e-mail: pierre-clement.blaud@imt-atlantique.fr, philippe.chevrel@imt-atlantique.fr, fabien.claveau@imt-atlantique.fr).*
** *IMT Atlantique, GEPEA, UMR CNRS 6144, F-44307 Nantes, France, (e-mail: pierrick.haurant@imt-atlantique.fr)*
*** *CEA, CEA Tech Pays de la Loire, F-44340 Bouguenais, France, (e-mail: anthony.mouraud@cea.fr)*

Abstract: This study aims to aid understanding of Model Predictive Control (MPC) alternatives through comparing most interesting MPC implementations. This comparison will be performed intrinsically and illustrated using the four-tank benchmark, widely studied by academics taking care of industrial perspectives. Although MPC provides advanced control solutions for a wide class of dynamical systems, challenges arise in managing the compromise between accuracy, computational cost and resilience, depending on the type of model used. In this study, linear, linear time-varying and non-linear MPCs are compared to MPC that uses a neural network based predictive model identified from data. The tuning and implementation methods considered are discussed, and accurate simulation results provided and analyzed. Precisely, the performance of each method (linear, linear time-varying, non-linear MPC) are compared to the neural MPC. Pros and cons of neural MPC are highlighted.

Keywords: Model predictive control, artificial neural network, quadruple tank process

1. INTRODUCTION

Among advanced control techniques, Model Predictive Control (MPC) has been widely used Mayne et al. (2000) and can control multi-inputs multi-outputs (MIMO) dynamical systems while handling state and input constraints. Expectations are specified by a quadratic cost function over a receding horizon, which is to be optimized. The main features are : it is a model based approach, involving a quadratic cost function, to manage control objectives and able to deal with hard constraints on key process signals Mayne et al. (2000). In classical MPC, the dynamical system is represented using a state-space model which can be linear Muske and Rawlings (1993) or non-linear Grüne and Pannek (2017). The cost function is optimized online, and the first sample of the control law computed over the receding horizon is applied to the actuators. This online implementation of MPC leads to tractability issues. Involving a linear model leads to a convex optimization problem, while with a non-linear model the optimization becomes non-convex requiring potentially intensive computation with the risk to converge to a local optimum Henrion and Lasserre (2004). Furthermore, finding the most suitable state space model is not easy, and may reveal challenging and time-consuming issues, as argued for instance in Maddalena et al. (2020). A trade-off must thus be found between model accuracy, time spent on finding a model and optimization efficiency.

The use of process input and output data within MPC has been investigated by practitioners to reduce modeling complexity and engineering time spent while increasing model accuracy. One example is to approach the process with a linear model with an added time-varying function Aswani et al. (2013) using past data to get a better fit. Finding the nominal linear state space model may however require engineering time. The alternative in Terzi et al. (2020) is to consider black-box model identification. They compared four black-box models, namely classical parametric models, with AutoRegressive eXogenous (ARX) and Output Error (OE) methods, then neural network based models with time-dependent Echo State Networks (ESN) and gradient-based Long Short-Term Memory (LSTM). It was found that using neural networks reduced modeling error compared to parametric models.

Among time-dependent or gradient-based neural networks, Radial Basis Function (RBF) based neural networks Li et al. (2019), ESN Armenio et al. (2019), Feedforward Neural Network (FNN) Kittisupakorn et al. (2009) and Recurrent Neural Networks (RNN) Wong et al. (2018), may be interesting for systems identification.

In Li et al. (2019), a non-linear chemical process is identified using RBF neural networks, then MPC controller is based on this RBF model. In Armenio et al. (2019),

⋆ This research was funded by Conseil Régional des Pays de la Loire.

10.1016/j.ifacol.2022.09.048

a non-linear process is identified with ESN. First, the authors analyze the stability properties of the obtained ESN model, which is then used by the MPC controller. In Kittisupakorn et al. (2009), a non-linear chemical process is identified using FNN, which is then used as a model within MPC. The authors compared the controller to a classical Proportional Integral (PI) controller and found that the approach outperformed the conventional PI controller. Within reference Wong et al. (2018), an RNN is used to identify a process from its non-linear state space model. Four RNN networks were trained and used for MPC design, and the best RNN-MPC control performance obtained was compared with the classical non-linear MPC based on a physical state space model, which showed close average performances.

The studies reported in Li et al. (2019); Armenio et al. (2019); Kittisupakorn et al. (2009); Wong et al. (2018) use the same non-linear physical model (known and calibrated) for data generation and MPC controllers. They thereby introduce a methodological bias ignoring the issue of engineering time spent to develop and calibrate the design model for control. Moreover, when an MPC controller comparison is provided, the NNMPC cannot perform better than the NLMPC, because the data used for NN-based model identification are generated from the non-linear model based on simplified physical equations. Even in the case for which more realistic data are considered to train a NN (e.g., Terzi et al. (2020)), no comparison is proposed between classical MPC and NNMPC.

The article revisits and illustrates for practitioners the alternatives regarding the implementation of the MPC principles. In particular, the alternatives in terms of choice of the prediction model are considered. The impact of the choice of one model or another on the quality of the synthesized MPC is questioned on different aspects: performance in terms of reference tracking versus computational complexity. To address this problem, four MPCs are developed and compared while considering the Quadruple tank process (QTP). The first controller is the linear model-based MPC (LMPC). For the second, MPC relies on a linear time-varying model (LTV-MPC), where for each iteration, the model is linearized from a nonlinear model. Third, a nonlinear model MPC (NLMPC) is considered as a third controller, while a neural network MPC (NNMPC) is considered fourth.

In summary, the main contributions of this work include:

- A fair and detailed comparison of four MPC-based control laws, from classical (linear, LTV and non-linear) ones to neural network MPC.
- The use of a detailed multi-physics model programmed with the Modelica language, to generate realistic data and allow a realistic simulation for comparison purposes.

The paper is organized as follows: Section 2 presents the considered MPC controllers. Section 3 details aspects of neural networks, while section 4 presents the process used to evaluate the considered MPC solutions. Section 5 compares the four MPC controllers law, before concluding in Section 7.

2. MODEL PREDICTIVE CONTROL FORMULATIONS

The purpose of MPC is to stabilize the system under control while optimally steering its states toward references. The quadratic cost function used in classical MPC allows compromising between the performance of the references tracking and control inputs penalty to drive the system. Moreover, the optimization problem includes constraints on states and inputs and the system dynamics. The MPC requires the following steps online: system state measurement, resolution of the optimal control problem to get the ad hoc control signal, and then application of the first sample of the signal as control input to the process (next sampling time) Grüne and Pannek (2017). Denoting by x_k^r and u_k^r the reference trajectory associated to x_k, u_k the controlled system states and inputs, it is possible to define the deviation of states and inputs to the reference trajectory as: $\tilde{x}_k = x_k - x_k^r$ and $\tilde{u}_k = u_k - u_k^r$. Moreover, the "ideal" state-space representation of the system is given as: $x_{k+1} = f(x_k, u_k)$, where, x_k and u_k represent the sampling of the state and input variables at current time k, and this work considers different MPC implementations using different approximations.

Implementation 1 (LMPC) uses a linear model approximation: $\hat{x}_{k+1} = f_L(x_k, u_k) = A_d x_k + B_d u_k$, where A_d and B_d represent state and input matrices. Implementation 2 (LTV-MPC) uses the linear time-varying approximation : $\hat{x}_{k+1} = f_{LTV}(x_k, u_k) = A_k^{LTV} x_k + B_k^{LTV} u_k$, where A_k^{LTV} and B_k^{LTV} are the Jacobian matrices of the nonlinear system linearized around the current state and input Murillo et al. (2016):

$$A_k^{LTV} = \frac{\partial f_{NL}}{\partial x}|_{x(t),u(t)} \ , \ B_k^{LTV} \quad = \frac{\partial f_{NL}}{\partial u}|_{x(t),u(t)} \quad (1)$$

Implementation 3 (NL-MPC), f is approximated using the non-linear function : $\hat{x}_{k+1} = f_{NL}(x_k, u_k)$. Finally, in implementation 4 (NN-MPC), f is approximated using a neural network : $\hat{x}_{k+1} = f_{NN}(x_k, u_k)$. As in Grüne and Pannek (2017), the MPC is:

$$\min_{\tilde{u},\tilde{x}} \ \tilde{x}_N^T P \tilde{x}_N + \sum_{k=0}^{N-1} \tilde{x}_k^T Q \tilde{x}_k + \tilde{u}_k^T R \tilde{u}_k \quad (2)$$

$$\text{s.c. } x_{k+1} = f(x_k, u_k) \quad (3)$$

$$\tilde{x}_k = x_k - x_k^r \quad (4)$$

$$\tilde{u}_k = u_k - u_k^r \quad (5)$$

$$x_0 = x_m(t) \quad (6)$$

$$x_k \in \mathcal{X} \quad (7)$$

$$u_k \in \mathcal{U} \quad (8)$$

where Eq.2 is the quadratic cost function used with N time horizon and Q and R weighting matrices (semi-positive definite) of appropriate dimensions, also upper T denotes the matrix transpose. Eq.3 is the state equation of the dynamical model, Eq.4 the state deviation with the state reference, Eq.5 the input deviation with the input reference, Eq.6 the state measure from real plant at time t while Eq.7 and Eq.8 are the state and input constraints respectively.

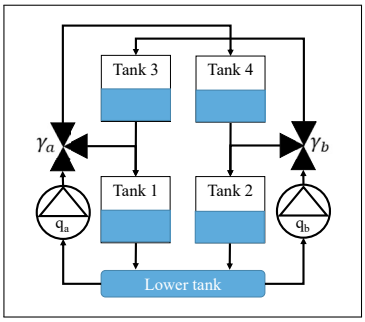

Fig. 1. QTP illustrative example.

3. NEURAL NETWORKS MODELING F_{NN}

Feedforward neural networks (FNNs) are the most common artificial neural network architecture. They were chosen for this reason, in order to allow for later comparisons. They include an input layer, one or more hidden layers (with several artificial neurons) and an output layer. They can approximate a large class of static non-linear functions. Even with only one hidden layer (with a finite number of neurons) and sigmoid activation function, FNNs are universal approximators Kim et al. (2018). In this study, FNNs are used to model one ahead predictor. The FNN inputs are the state and input of the dynamical system at current time k and the output is the predicted state at time k+1. The predicted output of the FNN with one hidden layer is equal to:

$$\hat{x}_{k+1} = \sigma^2(W^2(\sigma^1(W^1\sigma^0(W^0 v_k + b^0) + b^1)) + b^2) \quad (9)$$

with $\hat{x}_{k+1}$ the predicted output, σ^i the activation function and W^i the weight matrix. The input of the FNN v_k is the concatenation of current input u_k and state x_k : $v_k^T = cat(u_k^T, x_k^T)$. b^i known as biases. Exponent ^{0} denotes the input layer, exponent ^{1} the hidden layer and exponent ^{2} the output layer.

4. QUADRUPLE TANK PROCESS

4.1 Description

Without being an industrial process, Quadruple Tank Process (QTP) is a system which enables highlighting many difficulties encountered in controlling complex industrial processes such as, nuclear steam generation (Fig.1). As a result, it constitutes an excellent control benchmark for methodological research. QTP is non-linear and exhibits coupling between subsystems and its state variables and inputs must respect constraints Alvarado et al. (2011).

QTP will be used in the following to illustrate pros and cons of the four MPC implementations considered in this paper. The QTP process includes four tanks, two pumps and two three-way valves, where water flows from pump q_a to tanks 1 and 4 and from pump q_b to tanks 2 and 3. Each tank has an orifice which leaks liquid according to the water pressure: from tank 3 to tank 1, from tank 4 to tank 2, from tank 1 and from tank 2 to a lower tank outside the system. The control objective is to keep water levels within the four tanks at given references, due to appropriate pumps with flow necessarily greater than zero.

4.2 Physical model

A simplified non-linear equation involving instead of can be derived using Torricelli's law of water flow through an orifice Alvarado et al. (2011) :

$$\dot{h}_1(t) = \frac{-a_1}{S_c}\sqrt{2gh_1(t)} + \frac{a_3}{S_c}\sqrt{2gh_3(t)} + \frac{\gamma_a q_a(t)}{3600 S_c} \quad (10)$$

$$\dot{h}_2(t) = \frac{-a_2}{S_c}\sqrt{2gh_2(t)} + \frac{a_4}{S_c}\sqrt{2gh_4(t)} + \frac{\gamma_b q_b(t)}{3600 S_c} \quad (11)$$

$$\dot{h}_3(t) = \frac{-a_3}{S_c}\sqrt{2gh_3(t)} + \frac{(1-\gamma_b) q_b(t)}{3600 S_c} \quad (12)$$

$$\dot{h}_4(t) = \frac{-a_4}{S_c}\sqrt{2gh_4(t)} + \frac{(1-\gamma_a) q_a(t)}{3600 S_c} \quad (13)$$

h_1, h_2, h_3 and h_4 represent the water levels within the dedicated tanks, a_1, a_2, a_3 and a_4 the orifice areas of tanks 1 to 4 and S_c the cross section of the tanks. q_a and q_b are the pumps flow while γ_a and γ_b are the ratios of the three-way valve opening, Tab.1. The linearized models involving and are also provided in Alvarado et al. (2011):

$$\dot{h}_1(t) = \frac{-a_1 h_1(t)}{S_c\sqrt{\frac{2h_1^0}{g}}} + \frac{a_3 h_3(t)}{S_c\sqrt{\frac{2h_3^0}{g}}} + \frac{\gamma_a q_a(t)}{3600 S_c} \quad (14)$$

$$\dot{h}_2(t) = \frac{-a_2 h_2(t)}{S_c\sqrt{\frac{2h_2^0}{g}}} + \frac{a_4 h_4(t)}{S_c\sqrt{\frac{2h_4^0}{g}}} + \frac{\gamma_b q_b(t)}{3600 S_c} \quad (15)$$

$$\dot{h}_3(t) = \frac{-a_3 h_3(t)}{S_c\sqrt{\frac{2h_3^0}{g}}} + \frac{(1-\gamma_b) q_b(t)}{3600 S_c} \quad (16)$$

$$\dot{h}_4(t) = \frac{-a_4 h_4(t)}{S_c\sqrt{\frac{2h_4^0}{g}}} + \frac{(1-\gamma_a) q_a(t)}{3600 S_c} \quad (17)$$

h_i^0 denotes the water level used for linearization. It is important to note that these equations fail to efficiently model water levels when they are lower than 0.2m due to eddy effects Alvarado et al. (2011), and pumps dynamics are also neglected.

Table 1. Paramters of QTP Alvarado et al. (2011).

Parameters	Value	Unit	Description
$h_{1_{max}}$, $h_{2_{max}}$	1.36	m	Maximum water level
$h_{3_{max}}$, $h_{4_{max}}$	1.30	m	Maximum water level
h_{min}	0.2	m	Minimum water level
$q_{a_{max}}$	3.26	m^3/h	Maximum water flow
$q_{b_{max}}$	4	m^3/h	Maximum water flow
q_{min}	0	m^3/h	Minimum water flow
S_c	0.06	m^2	Tanks cross-section
a_1	1.31^{-4}	m^2	Surface of the leakage orifice
a_2	1.51^{-4}	m^2	Surface of the leakage orifice
a_3	9.27^{-5}	m^2	Surface of the leakage orifice
a_4	8.82^{-5}	m^2	Surface of the leakage orifice
γ_a, γ_b	0.3, 0.4	-	Three way valve opening
g	9.81	m/s^2	Gravitational acceleration

4.3 Fine-scale simulation

The QTP process is implemented for simulation using the equation-based object-oriented modeling language Modelica. In this case, the process considers water properties, pipes and pumps inertia are considering by using the Modelica standard library and pumps using the Building

library, the components are graphically assembled to build the QTP simulation model. An overflow outlet is added to the four tanks to consider the maximal water level and thereby avoid simulation breaking. The Modelica simulation has 1,049 equations and unknown variables. These equations could be used for the MPC controller; however, the simulation has some if-then-else conditions which make it challenging to use with the MPC controller, and thus a hybrid MPC controller is mandatory and leads to complicated analyses, design and optimization techniques Lazar (2006). Furthermore, the 1,049 equations programmed in the Modelica language need to be translated to specific modeling languages for mathematical optimization, and to the authors' knowledge there is no automatic tool for this purpose.

4.4 Data generation

Modelica was used to generate data using Dymola. The inputs (both pumps) are excited with signals and water levels which are displayed. The type of the input signals is important for dynamical system identification. Pseudo-random binary sequences (PRBS) was used to excite system with a widespread frequency spectrum. In our case, PRBS signals q_a and q_b are shown in Tab.2. The tanks' water levels are then saved at each 5s time interval. During data generation with PRBS, it appears that all water levels are never in a steady state. To remedy this issue and allow FNN to learn the plant steady state, a second kind of input signals is used (see Tab.2) which consists of a random value of pump command from minimum to maximum values.

Table 2. Parameters of input signals.

Parameters	PRBS	Constant	Unit
Time period	500	1200	s
Start time	500	30	s
End time	30	450	Days
Amplitude	q_{min} or q_{max}	q_{min} to q_{max}	m^3/h

5. COMPARISON OF THE 4 MPC CONTROL LAWS

5.1 FNN implementation

The FNNs were implemented within the Julia programming language. It is challenging to choose the correct numbers of hidden layers and neurons, and thus a program was implemented to train several FNNs. For instance, the hidden layers from 1 to 3 and neurons from 5 to 15 with 5 step neurons were considered. The "Swish" activation function was empirically chosen in this work, $f(x) = \frac{x}{1+e^x}$, Ramachandran et al. (2018). All FNNs were trained, and the one corresponding with the lowest loss function was selected. Training computations were performed using GPUs to quicken training with backpropagation algorithm. The optimizer used during training is Rectified Adam Liu et al. (2020), the batch size comprised 128 samples and 100 epochs were executed.

The simulated data fed the FNNs. Data are randomly separated into two parts: 80% for training and 20% for testing. The training data are further separated into two parts, where 90% feeds the FNN and 10% validate the data. This data division is executed at each 20 epochs, and the performance validation of the output neural network is evaluated using the fidelity measure, considered in this study as the Mean Squared Error (MSE) Wang and Bovik (2009): $MSE = \frac{1}{Nb}\sum_{i=1}^{Nb}[\hat{y}_i - y_i]^2$, with Nb the number of samples, $\hat{y}$ the output of the neural network and y the targeted data.

5.2 MPC implementation

LMPC uses the linear model defined by f_L (state equation) and Eq.14-17 to control the plant, and its linearizing at the nominal operating point: $h_1^0 = 0.65m$, $h2^0 = 0.66m$, $h_3^0 = 0.65m$ and $h_4^0 = 0.66m$. The LTV-MPC dynamical model f_{LTV} uses Eq.14-17, and the linearizing point is modified during each MPC execution using the current state and input. The discrete linear models used for LMPC and LTV-MPC are derived via exact discretization using a Zero-Order Hold (ZOH) at a sample period equal to 5s. In contrast, the NLMPC uses nonlinear model defined by f_{NL} and Eq.10-13 and using the Runge-Kutta method for discretization. The terminal weight matrix P (see eq. 2) is derived from the Lyapunov discrete equation: $P - A_d^T P A_d = Q$ where A_d the state matrix of the discrete model is linearized at the current state measure for LTV-MPC, NNMPC and NLMPC. The state weighting matrix Q is taken as $Q = \mathrm{diag}(1, 1, 1, 1)$ while the inputs weighting matrix $R = \mathrm{diag}(0.01, 0.01)$. N is the horizon length with 5 samples and the sampling time is equal to $5s$. The considered set-points were taken from Alvarado et al. (2011) and the water levels and corresponding pump flows of tanks 3 and 4 are derived using the reference values for tanks 1 and 2. Which is achieved using the static model with zero derivative, and it leads to the tank references shown in Tab.3.

Table 3. References at steady state.

Reference	Set-points	States computed	Inputs computed	Times (s)
1	$x_1 = 0.650$	$x_3 = 0.652$	$u_1 = 1.637$	0 to
	$x_2 = 0.650$	$x_4 = 0.664$	$u_2 = 1.988$	3000
2	$x_1 = 0.300$	$x_3 = 0.301$	$u_1 = 1.112$	3000 to
	$x_2 = 0.300$	$x_4 = 0.306$	$u_2 = 1.351$	6000
3	$x_1 = 0.500$	$x_3 = 0.305$	$u_1 = 2.201$	6000 to
	$x_2 = 0.750$	$x_4 = 1.200$	$u_2 = 1.361$	9000
4	$x_1 = 0.900$	$x_3 = 1.062$	$u_1 = 1.528$	9000 to
	$x_2 = 0.750$	$x_4 = 0.579$	$u_2 = 2.539$	12000

5.3 Fine simulation of the instrumented plant

Fine-scale plant modeling was exported using the Functional Mock-up Interface (FMI) from Dymola to be simulated using software. Moreover, the MPCs were implemented using JuMP package within Julia and the Ipopt optimization solver, which can solve quadratic programming, linear and non-linear. During simulation, the Functional Mock-up Unit (FMU) is simulated, and at each MPC execution (5s), the simulation is paused, the states are read by Julia program, the MPC is computed, the inputs control are sent to the actuator within the FMU and the simulation is run until the next execution. In addition, the states and input variables from the FMU are saved in a text file during the simulation, which allows evaluating and comparing each MPC solution presented in this work on the same basis.

5.4 Numerical results

Tab.4 shows the results obtained for the trained FNN and highlights that FNN 8 provides the lowest loss and it is selected within the NNMPC tuning.

Table 4. List of FNN trained for system identification (selected in bold).

FNN	Neurons per hidden layers	Hidden layers numbers	MSE_{train}	MSE_{test}
1	5	1	4.421×10^{-5}	4.420×10^{-5}
2	10	1	5.562×10^{-6}	5.572×10^{-6}
3	15	1	4.161×10^{-6}	4.186×10^{-6}
4	5	2	2.839×10^{-5}	2.829×10^{-5}
5	10	2	6.070×10^{-6}	6.082×10^{-6}
6	15	2	1.021×10^{-5}	1.022×10^{-5}
7	5	3	1.531×10^{-5}	1.529×10^{-5}
8	10	3	2.143×10^{-6}	2.174×10^{-6}
9	15	3	1.090×10^{-5}	1.091×10^{-5}

Fig.2 depicts the simulation results with h1 and h2 water levels controlled by LMPC, LTV-MPC, NLMPC and NNMPC. For all controllers, the regulated water levels stay within the states constraints, and h1 and h2 remain constant during steady state (see Fig.2.a and Fig.2.b). In addition, when the set-point changes, the water level overshoots are visible for all controllers (Fig.2.a at 6000s with h1 and Fig.2.b at 9000s with h2). Moreover, the QTP exhibits a non-minimum phase behavior (Fig.2.a see h1 at 9000s).

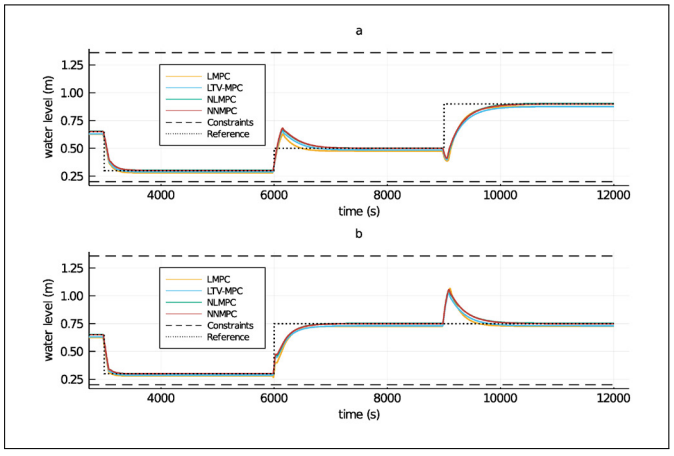

Fig. 2. Simulation results: water levels over time. (a) h1 water level. (b) h2 water level.

Fig.3 depicts the simulation results with h3 and h4 water levels controlled by LMPC, LTV-MPC, NLMPC and NNMPC. Once again, water levels h3 and h4 stay within the states constraints.

Fig.4 depicts the state trajectory with water levels h_1 and h_2 controlled by LMPC, LTV-MPC, NLMPC and NNMPC. Fig.4 highlights whether h_1 and h_2 reached their references and their trajectory. The h_1 and h_2 controlled by LMPC and LTV-MPC did not reach their references, unlike NLMPC and NNMPC. Furthermore, the h_1 and h_2 trajectory obtained with NLMPC and NNMPC are similar and feature only small differences when a sudden change occurs. The states trajectories obtained with LMPC and LTV-MPC also differ, with a better result for LTV-MPC yet significantly worse than that of NNMPC.

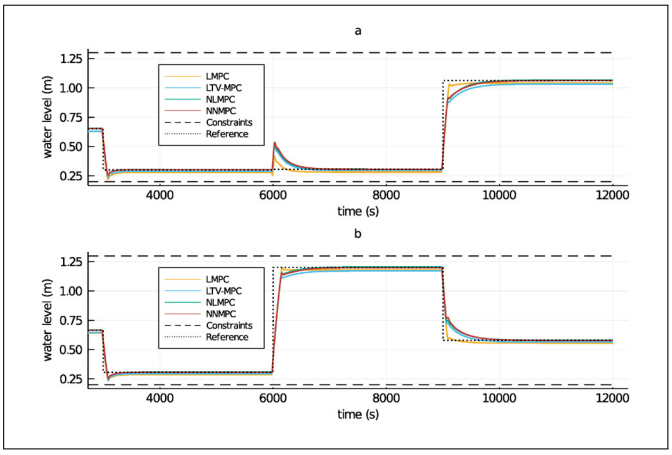

Fig. 3. Simulation results: water level over time. (a) h3 water level. (b) h4 water level.

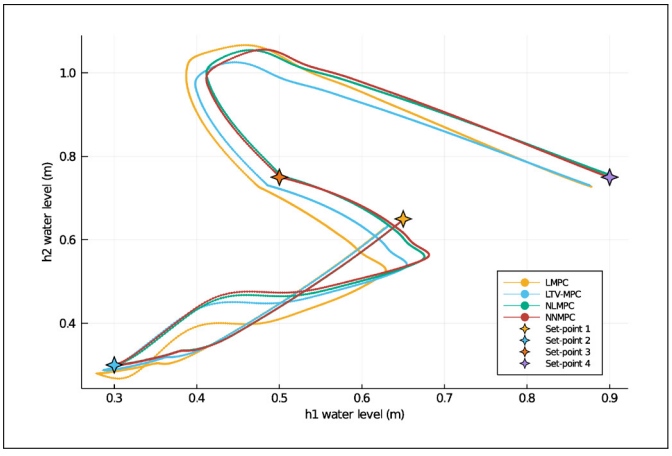

Fig. 4. h_2 over h_1 states trajectory with controllers.

Fig.5 depicts the state trajectory with water levels h_3 and h_4 controlled by LMPC, LTV-MPC, NLMPC and NNMPC. Fig.5 highlights whether h_3 and h_4 reached set-points and the chosen path. Also, h_3 and h_4 controlled by LMPC and LTV-MPC did not reach any set-point, while h_3 and h_4 controlled by NLMPC and NNMPC reached all set-points. Furthermore, the h_3 and h_4 trajectories from NLMPC and NNMPC are similar with small differences when a sudden change occurs. In contrast, the states trajectories from LMPC and LTV-MPC are different except for a relative proximity at steady state (see Table 5).

Table 5. Average steady-state absolute error and computation time of controllers.

Average	LMPC	LTV-MPC	NLMPC	NNMPC
Steady-state error (mm)	22.189	17.911	2.720	0.836
Computation time (s)	1.169	1.175	3.018	2.482

Tab.5 depicts the average steady state errors and the average computation time. It is noteworthy that the average computation time is equal to 2.482s from NNMPC, 3.018s

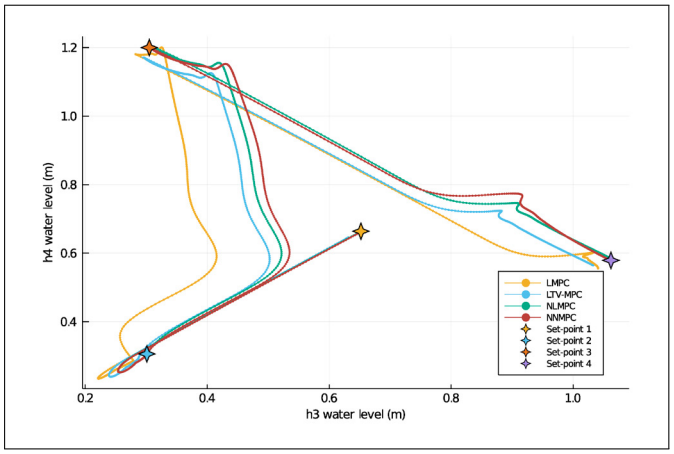

Fig. 5. h_4 over h_3 states trajectory with controllers.

for the NLMPC, 1.175s for the LTV-MPC and 1.169s to LMPC. It is remarkable that the NNMPC reduces computation time by 0.536s compared to the NLMPC, despite the latter's use of simplified NL model. While the average steady state error is equal to 0.836mm for NNMPC, 2.720mm for NLMPC, 17.911mm for LTV-MPC and 22.189mm for LMPC. Here, the lowest average steady state error is performed by NNMPC followed by NLMPC.

6. CONCLUSIONS

This study aimed at achieving a better view of MPC alternatives by comparing the most interesting MPC implementations: LMPC using a linear model, LTV-MPC using a linear time-varying model, NLMPC using a simplified nonlinear model and finally NNMPC using a neural network based model. The implementations were quantitatively compared when controlling the QTP. The results are good for all implementations, despite multi-variable coupling and non-minimum-phase zero. But the most interesting lies in the analysis of the differences. The results show that the NNMPC and NLMPC outperformed the LTV-MPC and LMPC controls, regarding in particular the steady state error. This advantage would be even more marked if the process were more highly non-linear. In the QTP case, the linear MPC may be considered competitive given their low cost of implementation, taking care therefore to penalize the tracking error integral. The perspectives of choosing less conventional artificial neural networks, but better suited to the problem of identifying, deserves to be deepened.

REFERENCES

Alvarado, I., Limon, D., Muñoz de la Peña, D., Maestre, J., Ridao, M., Scheu, H., Marquardt, W., Negenborn, R., De Schutter, B., Valencia, F., and Espinosa, J. (2011). A comparative analysis of distributed mpc techniques applied to the hd-mpc four-tank benchmark. *Journal of Process Control*, 21(5), 800–815. doi: 10.1016/j.jprocont.2011.03.003.

Armenio, L.B., Terzi, E., Farina, M., and Scattolini, R. (2019). Model predictive control design for dynamical systems learned by echo state networks. *IEEE Control Systems Letters*, 3(4), 1044–1049. doi: 10.1109/LCSYS.2019.2920720.

Aswani, A., Gonzalez, H., Sastry, S.S., and Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. *Automatica*, 49(5), 1216–1226. doi: 10.1016/j.automatica.2013.02.003.

Grüne, L. and Pannek, J. (2017). *Nonlinear model predictive control*, 45–69. Springer International Publishing, Cham. doi:10.1007/978-3-319-46024-6_3.

Henrion, D. and Lasserre, J.B. (2004). Solving nonconvex optimization problems. *IEEE Control Systems Magazine*, 24(3), 72–83. doi:10.1109/MCS.2004.1299534.

Kim, K.K.K., Patrón, E.R., and Braatz, R.D. (2018). Standard representation and unified stability analysis for dynamic artificial neural network models. *Neural Networks*, 98, 251–262. doi: 10.1016/j.neunet.2017.11.014.

Kittisupakorn, P., Thitiyasook, P., Hussain, M., and Daosud, W. (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19(4), 579–590. doi: 10.1016/j.jprocont.2008.09.003.

Lazar, M. (2006). *Model predictive control of hybrid systems: stability and robustness*. Ph.D. thesis, Technische Universiteit Eindhoven. doi:10.6100/IR612103.

Li, S., Jiang, P., and Han, K. (2019). Rbf neural network based model predictive control algorithm and its application to a cstr process. In *2019 Chinese Control Conference (CCC)*, 2948–2952. doi: 10.23919/ChiCC.2019.8865797.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J., Kingma, D.P., and Ba, J. (2020). On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*.

Maddalena, E.T., Lian, Y., and Jones, C.N. (2020). Data-driven methods for building control — a review and promising future directions. *Control Engineering Practice*, 95, 104211. doi:10.1016/j.conengprac.2019.104211.

Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814. doi: 10.1016/S0005-1098(99)00214-9.

Murillo, M., Sánchez, G., and Giovanini, L. (2016). Iterated non-linear model predictive control based on tubes and contractive constraints. *ISA Transactions*, 62, 120–128. doi:10.1016/j.isatra.2016.01.008.

Muske, K.R. and Rawlings, J.B. (1993). Model predictive control with linear models. *AIChE Journal*, 39(2), 262–287. doi:10.1002/aic.690390208.

Ramachandran, P., Zoph, B., and Le, V.Q. (2018). Searching for activation functions. In *International Conference on Learning Representations (ICLR)*.

Terzi, E., Bonetti, T., Saccani, D., Farina, M., Fagiano, L., and Scattolini, R. (2020). Learning-based predictive control of the cooling system of a large business centre. *Control Engineering Practice*, 97, 104348. doi: 10.1016/j.conengprac.2020.104348.

Wang, Z. and Bovik, A.C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1), 98–117. doi: 10.1109/MSP.2008.930649.

Wong, W.C., Chee, E., Li, J., and Wang, X. (2018). Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11). doi:10.3390/math6110242.